# MKFIFO

Vulnerable to TOCTOU issues

Sean Barnum, Cigital, Inc. [vita[1]]

2007-03-29

## Part "Original Cigital Coding Rule in XML"

Mime-type: text/xml, size: 7126 bytes

| | |
|---|---|
| **Attack Category** | • Path spoofing or confusion problem |
| **Vulnerability Category** | • Indeterminate File/Path<br>• TOCTOU - Time of Check, Time of Use |
| **Software Context** | • File Creation<br>• File I/O |
| **Location** | • sys/stat.h |
| **Description** | mkfifo makes a FIFO special file with name pathname. mode specifies the FIFO's permissions. It is modified by the process's umask in the usual way: the permissions of the created file are (mode & ~umask).<br>A FIFO special file is similar to a pipe, except that it is created in a different way. Instead of being an anonymous communications channel, a FIFO special file is entered into the file system by calling mkfifo.<br><br>Once you have created a FIFO special file in this way, any process can open it for reading or writing, in the same way as an ordinary file. However, it has to be open at both ends simultaneously before you can proceed to do any input or output operations on it. Opening a FIFO for reading normally blocks the file until some other process opens the same FIFO for writing, and vice versa. See fifo(4) for non-blocking handling of FIFO special files.<br><br>mkfifo() is vulnerable to classic TOCTOU attacks.<br><br>A call to mkfifo() should be flagged if the first argument (the file name) is used previously in a check. |
| **APIs** | **Function Name**: mkfifo — **Comments**: use |
| **Method of Attack** | The key issue with respect to TOCTOU vulnerabilities is that programs make assumptions about atomicity of actions. It is assumed that checking the state or identity of a targeted resource |

APIs:

| Function Name | Comments |
|---|---|
| mkfifo | use |

1. http://buildsecurityin.us-cert.gov/bsi-rules/35-BSI.html (Barnum, Sean)

| | |
|---|---|
| | followed by an action on that resource is all one action. In reality, there is a period of time between the check and the use that allows either an attacker to intentionally or another interleaved process or thread to unintentionally change the state of the targeted resource and yield unexpected and undesired results.<br><br>The mknod() call is a use-category call, which when preceded by a check-category call can be indicative of a TOCTOU vulnerability.<br><br>A TOCTOU attack in regards to mkfifo() can occur, for example, when<br><br>a. A check for the existence of a filename (check call) occurs<br><br>b. mkfifo() is executed<br><br>Between a and b, an attacker could, for example, create a hard link to the pathname, resulting ultimately in a "different" pipe being created than what was expected. |
| **Exception Criteria** | |
| **Solutions** | |

| **Solution Applicability** | **Solution Description** | **Solution Efficacy** |
|---|---|---|
| Generally applicable to all mkfifo() calls. | Utilize a file descriptor version of check and use functions, if possible. | Effective. |
| Generally applicable to all mkfifo() calls. | The most basic advice for TOCTOU vulnerabilities is to not perform a check before the use. This does not resolve the underlying issue of the execution of a function on a resource whose state and identity cannot be assured, but it does help to limit the false sense of security given by the check. | Does not resolve the underlying vulnerability but limits the false sense of security given by the check. |

| | | | |
|---|---|---|---|
| | Generally applicable to all mkfifo() calls. | Limit the interleaving of operations on files from multiple processes. | Does not eliminate the underlying vulnerability but can help make it more difficult to exploit. |
| | Generally applicable to all mkfifo() calls. | Limit the spread of time (cycles) between the check and use of a resource. | Does not eliminate the underlying vulnerability but can help make it more difficult to exploit. |
| | Generally applicable to all mkfifo() calls. | Recheck the resource after the use call to verify that the action was taken appropriately. | Effective in some cases. |

| | |
|---|---|
| **Signature Details** | int mkfifo(const char *pathname, mode_t mode); |
| **Examples of Incorrect Code** | `/* Check added */`<br><br>`#include "sys/types.h"`<br>`#include "sys/stat.h"`<br><br>`int status;`<br>`int check_status;`<br>`struct stat statbuf;`<br>`...`<br>`check_status=stat("/home/cnd/mod_done", &statbuf);`<br><br>`...`<br>`status = mkfifo("/home/cnd/mod_done", S_IWUSR \| S_IRUSR \| S_IRGRP \| S_IROTH);` |
| **Examples of Corrected Code** | `/* No check needed */`<br><br>`#include "sys/types.h"`<br>`#include "sys/stat.h"`<br><br>`int status;`<br>`...`<br>`status = mkfifo("/home/cnd/mod_done", S_IWUSR \| S_IRUSR \| S_IRGRP \| S_IROTH);` |
| **Source References** | • Viega, John & McGraw, Gary. *Building Secure Software: How to Avoid Security Problems* |

| | |
|---|---|
| | *the Right Way*. Boston, MA: Addison-Wesley Professional, 2001, ISBN: 020172152X, pp 222<br>• UNIX man page for mkfifo()<br>• http://www.die.net/doc/linux/man/man3/mkfifo.3.html<br>• GNU Core Utilities race condition file-permissions vulnerability[3].<br>• The IEEE and The Open Group. "mkfifo - make a FIFO special file[4]." The Open Group Base Specifications Issue 6; IEEE Std 1003.1, 2004 Edition (2004). |
| **Recommended Resource** | |
| **Discriminant Set** | **Operating System**: • UNIX<br>**Languages**: • C • C++ |

## Cigital, Inc. Copyright

1. mailto:copyright@cigital.com